UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/556,396 | 10/18/2006 | Shigenori Doi | 2005_1755A | 6455 |

52349 7590 11/05/2010

WENDEROTH, LIND & PONACK L.L.P.
1030 15th Street, N.W.
Suite 400 East
Washington, DC 20005-1503

| EXAMINER | |
|---|---|
| VU, TUAN A | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2193 | |

| NOTIFICATION DATE | DELIVERY MODE |
|---|---|
| 11/05/2010 | ELECTRONIC |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

ddalecki@wenderoth.com
eoa@wenderoth.com

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/556,396 | DOI ET AL. | |
| | Examiner | Art Unit | |
| | TUAN A. VU | 2193 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *10/18/06*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *10 November 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *9/15/09; 2/05/10*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed 10/18/2006.

Claims 1-20 have been submitted for examination.

***Claim/Specifications Objections***

2. Claims 1-2, 4-8, 10, 12, 15, 18-20 are objected to because of the following informalities: some lines (or group of words) do not have proper spacing between words; e.g. claim 1, li. 4, 14; clm 2, li. 6; clm 4, li. 4; clm 5, li.6; clm 6, li. 5; clm 7, li. 15; clm 8, li. 3; clm 10, li. 10; clm 12, li. 5; clm 15, li. 4; clm 18, li. 4; clm 19, li. 4; clm 20, li. 5. Appropriate correction is required. Some portions of the Specifications exhibit the same absence of spacing between words; correction is strongly urged.

***Claim Rejections - 35 USC § 101***

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 1-17, 19 rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The current focus of the Patent Office in regard to statutory inventions under 35 U.S.C. § 101 for method claims and claims that recite a judicial exception (software) is that the claimed invention recite a practical application. The following link on the World Wide Web is the United States Patent And Trademark Office (USPTO) reference in terms of guidelines on a proper analysis on 35 U.S.C. §101 rejection.

<http://www.uspto.gov/web/offices/pac/dapp/opla/preognotice/guidelines101_20051026.pdf>

Specifically, claims 1 and 19 recite 'execution control device' comprising judging unit, first unit to instruct a processor to execute and compile, second unit to instruct the processor to execute native code, third unit to instruct the processor to compile in parallel with execution of said first unit and second unit. According to the Specifications, the first, second, third and judging unit are software implemented entities to judge significance of indicator and manage or control priority of queue. The listing of software per se (i.e. Functional Descriptive Material in software arts) as perceived has no hardware to support realization of software functionalities referred to as *judging unit, first, second and third unit* and cannot permit one to categorize the software functionalities as any category of subject matter.

The reciting of "control device' fail to explicitly include 'processor' being integral to the 'device'; hence, claims 1, 19 amount to mere "Functional Descriptive Material" (see Guidelines Annex IV, pg. 52-54) without explicit inclusion of real-world tangible apparatus cannot be construed as belonging to any of the four categories of permissible subject matter (not a apparatus, a process, article of manufacture, or composition of matter) and are rejected for constituting non-statutory subject matter.

As 'storage unit' having 'register compilation information' is not recited as being part of the 'execution control device' of the base claims, dependent claims 2-17 fail to remedy to the lack of hardware in claim 1, whereas are also rejected as non-statutory subject matter.

***Claim Rejections - 35 USC § 103***

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are

> such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-12, 14-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Holzle et al, USPN: 6,240,548 (herein Holzle, with *incorporated by reference*: 08/944334 or USPN 5,995,754 – herein **Holzle2**), in view of APA (Admitted Prior Art - Specifications: Background of Invention: pg. 2).

**As per claim 1**, Holzle discloses a program execution control device for causing a processor to execute a program composed of one or more sets of bytecodes including a bytecode for invoking a bytecode set, comprising:

a judging unit operable, for each execution of an invocation bytecode during execution of the program, to judge whether a bytecode set targeted for invocation is already compiled (are to be compiled … if method is not compiled– col. 4 lines 13- 45; col. 2 lines 31-34, lines 51-53) into *native* code specific to the processor(machine instruction - col. 11 lines 48-58);

a first unit operable to instruct the processor so that the target bytecode set is sequentially interpreted and executed (e.g. method is not compiled … … bytecodes … are interpreted – col. 4 lines 40- 45), and to issue a request to compile the target bytecode set to native code(dynamically compiled byte codes – col. 3 li. 55-62; *compiler 642 → runtime 646* -Fig. 6);

a second unit operable, when the target bytecode set is judged to be compiled, to instruct the processor to execute native code resulting from the compilation (Fig 6 - Note: machine code - col. 1 lines 61-64 – reads on processor executing compiled bytecode or native code); and

a third unit operable to instruct the processor to compile a bytecode set indicated by a compilation request issued by said first unit (Fig. 6; *both interpreted and dynamically compiled* –

col .3 lines 55-57), in parallel with the bytecode interpretation and execution by said first unit (interpreter 148, compiler 50 … executed code 50 – col. 4 lines 34-61; Figure 1).

Holzle does not explicilty disclose second unit *issuing a request to compile* target bytecode into native code *when the target bytecode set is judged to be uncompiled*; nor does Holzle disclose third unit instructing (the processor for) compiling of bytecode (as indicated by first unit request) *in parallel* with bytecode interpretation/execution of first unit *as well as execution of native code* execution by said second unit.

Holzle discloses, instead of executing bytecodes using an interpreter, via compiler 642, converting byte codes into machines instructions (col. 11 lines 48-58; Fig. 6) for the runtime system of operating system 660 (machine code - col. 61 lines 61-64); hence has disclosed issuing a request for compiler to compile bytecode if bytecode is not yet in its native form. Converting bytecodes into machine instructions native to a computer of a specific operating system or architecture was a well-known concept allowing Java machine or JIT engine to efficiently manage runtime bytecodes and this is disclosed in APA (Specification, pg. 2) where some JIT to reduce overhead converts an invoked method into native code if it is judged that in its bytecode form, it has not been compiled. And converting using a JIT bytecode (for a method or class) loaded at runtime into its native code form. Based on the concurrent action of interpreting and dynamic compilation in Holzle's runtime (Fig. 1), it would have been obvious for one skill in the art at the time the invention was made to implement the runtime interpreter/compiler by Holzle so that if it judged that the targeted bytecode is not yet compiled, to send (by the first unit) *a request to* a runtime compiler – such as a JIT as in APA – in order to have(via the third unit) the targeted bytecode compiled (using the interpreter/compiler in Holzle's Fig. 6) into a native code

at runtime, and to have interpretation of bytecodes (by first unit) to be concurrent with the dynamic compilation and execution of compiled *native code* (e.g. using the second unit) as set forth above, because this concurrency of interpretation/execution with as need compilation as taught in APA would help to increase overall speed to execute targeted bytecode and/or reduce overhead of runtime, as set forth in APA in view of the dual capability (executing compiled code and selectively compiling) of just-in-time virtual machine.

**As per claim 2**, Holzle does not explicitly disclose wherein said program execution control device operates under control of a multitask operating system, the compilation by said third unit is executed as a separate task from the bytecode execution by said first unit and the native code execution by said second unit; where the tasks of said first and second units are assigned a higher priority level than a priority level assigned to the task of said third unit

Holzle disclose compilation as a task to monitor (col. 6 lines 23-32) for detecting a idle period to support usage of list of methods slated for compilation (execution list … queue - col. 12 line 25 to col 13 line 7) and prioritizing in terms of postponing the task of compiling (e.g. of candidate methods in the list) when determining that compiling would require too high an overhead to allow compilation (col. 6 lines 55-59) hence this is indicative a task prioritizing where the tasks of said first and second units are assigned a higher priority level than a priority level assigned to the task of said third unit based on some cost determination (col. 3 line 64 to col. 4 line 19), such that dynamic compilation (by third unit request) is delayed until pause can be detected (col. 5 lines 8-56) which is indicative that the intepretation/execution by second/first unit have priority until the runtime reach a pause or idle periods. Based on the monitoring for idle periods from above, it would have been obvious for one skill in the art at the time the

invention was made to implement the interpretation and dynamic compilation so that the compilation and interpreting/execution of bytecodes (as queued in the execution list) belong to a multitask operating system such that priority is given first to the interpreting (first unit) and execution native code (second unit) when it is determined that compilation of uncompiled bytecode is deemed taxing an overhead – as in APA and Holzle - over the runtime of the dynamic compilation, because muli-tasking and control in terms of prioritizing so to have the task of compiling(e.g. via request by the third unit) of bytecode postponed or alloted with lower priority than that of the task of interpreting/executing would help the overall performance of the interpretation/execution endeavor (e.g. see APA, Specifications: reduce overhead, pg. 2).

**As per claim 3**, Holzle disclose a switching unit operable to switch to task execution by said third unit when task execution by said first or second unit is placed in a standby state (col. 5 lines 45-56; Fig. 2; see rationale of claim 2).

**As per claim 4**, Holzle discloses said third unit instructs the processor to compile each bytecode set indicated by compilation request information registered in the storage unit, in parallel (refer to rationale in claim 1) with the bytecode interpretation and execution by said first unit as well as with the native code execution by said second unit;

but does not explicitly disclose a request management unit operable to *register compilation request information in a storage unit* in response to a compilation request issued by said first unit and *manage the registered compilation request information, each piece of compilation request information being used for compiling a bytecode set* indicated by a corresponding compilation request.

Holzle disclose inlined database (col. 6 lines 49) for maintaining of candidate list having been compiled and queueing of candidate methods in which some might be suppressed or added (col. 6 line 52 to col. 7 line 9) depending on threshold or count regarding compilation overhead information (see *incorporated by reference* **Holzle2**: Fig. 3-7). Hence the management of queue for priority determination with respect to elements being queued entails a form of registration (remove) or deregistration (add) of candidate methods; that is, the registered compilation information using database or threshold/count information related to the queued items being disclosed or obvious. Regarding the latter, it would have been obvious for one of ordinary skill in the art to implement the inlined database in conjunction with management of queued candidate methods as from above so that a registration unit supporting Holzle's compilation is operable from a storage unit including registered request information, the information maintained as count or threshold or overhead indication being used to support suppressing, compiling or updating the list of bytecode methods in the queueing approach by Holzle, because only under proper registration and dynamic reuse or inlining of registered bytecode-related information as taught in Holzle's database list (col. 9 line 1 to col. 10 line 31) and overhead metrics (see Holzle2) would each candidate methods and related request information be visible and timely supportive of the prioritization as taught in Holzle or Holze2's dynamic reduction of overhead ( for queued bytecode sets) with purport to alleviate runtime overhead via taking advantage of idle periods (col. 5 l. 20-56).

**As per claim 5**, Holzle does not explicitly disclose wherein said request management unit *places pieces of compilation request information in a queue in an order in which corresponding compilation requests are received*, and said third unit instructs the processor to

compile bytecode sets in order *starting from a bytecode set indicated by a first piece of queued* compilation request information.

But associating overhead and priority value based on execution list and inline database for associating candidate methods being queued as disclosed in Holzle (*priority value* - col. 8 line 66 to col. 10 line 31) to support prioritized compilation and updating of list and database entails that compilation request information are associated with the queue, because placing compilation request information in said queue having predefined "priority" value (i.e. order starting from a bytecode set indicated by a first piece of queued compilation request information ) would have been obvious in view of the obvious rationale as set forth in claim 4 from above, for the same reasons.

**As per claim 6**, Holzle does not explicitly disclose wherein said request management unit does not register compilation request information in duplicate, if compilation request information for a bytecode set indicated by a compilation request is already registered in the storing unit. But based on the removal of bytecodes already compiled, the concept of duplicated compilation request is disclosed. Hence, based on the approach of inlining database information and dynamic removing of queue elements as set forth in claims 4-5, not registering redundant or duplicated compilation request would have been obvious because database and properly managed queue cannot maintain duplicate therein.

**As per claim 7**, Holzle discloses:

a priority information acquiring unit operable to acquire information ( col. 12 lines 36-50) showing a priority level of each bytecode set, wherein said request management unit includes:

a specifying subunit operable, in response to a compilation request issued by said first unit, to specify with reference to the acquired priority information a priority level (col. 12 lines 25-56; col. 7 lines 30-67; Fig. 3 ) of a bytecode set.

Holzle does not explicilty disclose priority information indicated by the compilation request; but based on information and priority level as set forth above this request issuing regarding whether to compile or delay compilation of a bytecode set would have been obvious as per the rationale set forth in claim 1.

Nor does Holzle explicitly disclose:

a *comparing subunit operable to compare* the specified priority level with a priority level of each bytecode set indicated by queued compilation request information in the storage unit; and

a *determining subunit operable to determine a position for placing a new piece of compilation request information* for the bytecode set indicated by the compilation request, so that the *registered-pieces of compilation request information are queued in descending order* of priority.

Holzle disclose "execution list" and database list in terms of queueing (queue - col. 12 line 24 to col. 13 line 6) managed via adding and removing candidate method as compilation target based on priority level determination (e.g. highest first), so that the list is reevaluated and updated (col. 8 line 54 to col. 9 line 12) using priority value based on some factors (or counters) like frequency of execution, position within the list. It would have been obvious for one of ordinary skill in the art to implement the queue ordering and position updating, adding, removing methods in the ordered list/queue from above by Holzle so that a comparing unit compares priority level of each bytecode set as they are queued with respect to its priority and position

therein, and using a subunit to determine position for placing a new piece of request information so as to implement a managed order where *registered* pieces of compilation (refer to rationale in claims 4-5) request information respect a descending order starting from the highest priority as contemplated in Holzle (see col. 12 lines 24-30) because highest order compilation within a dual interpretation/compilation runtime capacity (see JIT in APA) enforces compilation precedence of what is needed over what is not (see Holzle: Fig. 3) as that would obviate unnecessary overhead cost as endeavored in both Holzle and APA.

**As per claim 8**, Holzle does not explicitly disclose

a relational information acquiring unit operable to acquire relational information showing each bytecode set together with all bytecode sets related to the bytecode set; and a detecting unit operable to detect, with reference to the relational information, any bytecode set related to the bytecode set indicated by the compilation request, wherein said request management unit registers compilation request information for the related bytecode set detected by said detecting unit.

But this *relational information* falls under the ambit of registering bytecode set with priority indicator for enabling priority-based compilation of target bytecode as set forth in claims 4-5; hence it would have been obvious for one of ordinary skill in the art to implement the database/execution list in Holzle's priority-based determination so that relational information unit is added for acquiring priority information related to the registering of the target bytecode in conjunction with a detecting unit for detecting and correlating said *relational information* with the registered bytecode in accordance to the queue management as set forth above (Holzle: Fig.

3; col. 12 line 24 to col. 13 line 6), because this would support Holzle's priority based compilation, for the same reasons mentioned in claims 4-5.

**As per claim 9**, Holzle does not explicitly disclose: a priority information acquiring unit operable to acquire information showing a priority level of each bytecode set; wherein with reference to the acquired priority information, said third unit instructs the processor to compile bytecode sets indicated by pieces of compilation request information registered in the storage unit, in descending order of priority.

But this falls under the ambit of the subject matter set forth in claims 7 and 8; and would have been obvious for the reasons set forth therein.

**As per claim 10**, Holzle discloses a count recording unit operable to keep a count of compilation requests made to a respective bytecode set when compilation of the bytecode set is repeatedly requested, and records the request count to the storage unit (col. 4 lines 41-50; queue of methods … counter – col. 6 lines 52-64; col. 12 lines 36-50) as part of compilation request information for the bytecode set; and based on using a threshold value (col 4 lines 51-61; col. 5 lines; col. 6 li. 52 to col. 7 line 8) as in Holzle's use of counters, it would have been obvious for one of ordinary skill in the art to implement the counter as set forth above so that an acquiring unit is implemented along so to be operable to acquire a threshold of request count, wherein said third unit instructs the processor to compile bytecode sets in order in which respective requests counts exceed the threshold, because the purpose of having a counter, as well-known in the computer arts, is for supporting a range or a limit as criterion for a action to be taken (see Holzle: col. 6 li. 52 to col. 7 line 8).

**As per claim 11**, Holzle discloses (by virtue of the rationale in claims 7 and 10) a count recording unit operable to keep a count of compilation requests made to a respective bytecode set when compilation of the bytecode set is repeatedly requested, and records the request count to the storage unit as part of compilation request information for the bytecode set; and an order altering unit operable to compare the respective request counts and alter positions of pieces of queued compilation request information in descending order of request count.

**As per claim 12**, Holzle discloses (by virtue of the rationale in claims 7 , 9 and 10) wherein said request management unit manages a plurality of queues with different priority levels, and said third unit instructs the processor to compile bytecode sets in order starting from bytecode sets indicated by compilation request information placed in a highest priority queue.

**As per claim 14**, Holzle does not explicitly disclose a second judging unit operable, when said judging unit judges that the target bytecode set is uncompiled, to judge whether the target bytecode set is currently under compilation; and a fourth unit operable, when the target bytecode set is judged to be currently under compilation, to wait until the compilation is done and to subsequently instruct the processor to execute native code resulting from the compilation.

But based on the successive loading of method bytecode and the pertinent as to whether it is in a native or uncompiled form (see claim 1; Holzle: Fig. 3, 4, 6; col. 10 lines 4-10), it would have been obvious for one of ordinary skill in the art to implement the first judging unit and third unit so as they are supported by second judging unit and fourth unit to effectuate the repeated process by which sequence of bytecode targeted for execution or compilation determination as taught in claim 1.

**As per claim 15**, Holzle discloses a request management unit operable to register compilation request information in a storage unit in response to a compilation request issued by said first unit and manage the registered compilation request information, each piece of compilation request information being used for compiling a bytecode set indicated by a corresponding compilation request (refer to *first unit* of claim 1, *register unit* of claim 4), wherein said third unit instructs the processor to compile each bytecode set indicated by compilation request information registered in the storage unit, in parallel with the bytecode interpretation and execution by said first unit as well as with the native code execution by said second unit (refer to claim 1).

**As per claim 16**, Holzle discloses a priority altering unit operable to temporarily raises a priority level of the compilation task, when the comparison shows that the priority level of the bytecode set is higher than the priority level of the instruction execution task (*may be removed ... a new highest priority ... is identified* – col. 8 lines 54-65; *updated ... re-evaluated* - col. 9 lines 1-12 – Note: re-evaluating and changing value for a short interval in runtime reads on temporarily altering)

and (by virtue of claim 7) discloses a priority information acquiring unit operable to acquire information *showing a priority level* of each bytecode set; a comparing unit operable to *compare priority levels of the bytecode set targeted for the compilation and of the instruction execution task*; and a priority altering unit operable to temporarily raises a priority level of the compilation task, when the *comparison* shows that the priority level of the bytecode set is higher than the priority level of the instruction execution task.

**As per claim 17**, Holzle discloses a request management unit operable to register compilation request informatio (refer to claim 4) in a storage unit in response to a compilation request issued by said first unit and manage the registered compilation request information, each piece of compilation request information being used for compiling a bytecode set indicated by a corresponding compilation request, wherein said third unit instructs the processor to compile each bytecode set indicated by compilation request information registered in the storage unit, in parallel with the bytecode interpretation and execution by said first unit as well as with the native code execution by said second unit (refer to claim 1).

**As per claim 18**, Holzle discloses program execution control method for causing a processor to execute a program composed of one or more sets of bytecodes including a bytecode for invoking a bytecode set, comprising:

a judging step, for each execution of an invocation bytecode during execution of the program, of judging whether a bytecode set targeted for invocation is already compiled to native code specific to the processor;

a first step, when the target bytecode set is judged to be uncompiled, of instructing the processor so that the target bytecode set is sequentially interpreted and executed, and of issuing a request to compile the target bytecode set to native code;

a second step, when the target bytecode set is judged to be compiled, of instructing the processor to execute native code resulting from the compilation; and

a third step of instructing the processor to compile a bytecode set indicated by a compilation request issued in said first step, in parallel with the bytecode interpretation and

execution in said first step as well as with the native code execution in said second step; all of which steps having been addressed in claim 1.

**As per claim 19**, Holzle discloses control program for causing a processor to execute another program composed of one or more sets of bytecodes including a bytecode for invoking a bytecode set, comprising:

a judging step, for each execution of an invocation bytecode during execution of the program, of judging whether a bytecode set targeted for invocation is already compiled to native code specific to the processor;

a first step, when the target bytecode set is judged to be uncompiled, of instructing the processor so that the target bytecode set is sequentially interpreted and executed, and of issuing a request to compile the target bytecode set to native code;

a second step, when the target bytecode set is judged to be compiled, of instructing the processor to execute native code resulting from the compilation; and

a third step of instructing the processor to compile a bytecode set indicated by a compilation request issued in said first step, in parallel with the bytecode interpretation and execution in said first step as well as with the native code execution in said second step; all of which having been addressed in claim 18.

**As per claim 20**, Holzle discloses a recording medium storing a control program for causing a processor to execute another program composed of one or more sets of bytecodes including a bytecode for invoking a bytecode set, said control program comprising:

a judging step, for each execution of an invocation bytecode during execution of the program, of judging whether a bytecode set targeted for invocation is already compiled to native code specific to the processor;

a first step, when the target bytecode set is judged to be uncompiled, of instructing the processor so that the target bytecode set is sequentially interpreted and executed, and of issuing a request to compile the target bytecode set to native code;

a second step, when the target bytecode set is judged to be compiled, of instructing the processor to execute native code resulting from the compilation; and

a third step of instructing the processor to compile a bytecode set indicated by a compilation request issued in said first step, in parallel with the bytecode interpretation and execution in said first step as well as with the native code execution in said second step; all of which considered addressed by virtue of the rationale and rejection in claim 1.

7. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Holzle et al, USPN: 6,240,548 (herein Holzle, with *incorporated by reference*: 08/944334 or USPN 5,995,754 – herein **Holzle2**), in view of APA (Admitted Prior Art - Specifications: Background of Invention: pg. 2) and further in view of Shaylor, USPubN: 2002/0104076 (herein Shaylor)

**As per claim 13**, Holzle discloses (by virtue of the rationale in claims 7 and 10) a special request information acquiring unit operable to acquire, prior to execution of the program, information showing a plurality of bytecode sets requested to be compiled, wherein said request management unit registers, compilation request information for all bytecode sets shown by the special request information, in a highest priority queue.

Holzle does not disclose special management unit registers compilation request in a batch.

Analogous to the concept of batch collection of request information, Holzle teaches task management with pre-collected database list, so that this is reusable inlined database for inlining the database list or counter database (col. 6 lines 34-51; **Holzle2**: col. 17 lines 37-46) into the priority-based target method bytecode compilation. It would have been obvious for one of ordinary skill in the art to implement the pre-collected amount of compilation information in view of the queue registering as set forth in claims 4-5, so that the inlining is implemented as a special batch request, a traditional approach which according to Shaylor (see Shaylor: para 0012, pg. 1 ), requires that extensive knowledge about classes be known at compile time, the batch including all the pre-collected dababase information as set forth above, because this batch form or pre-bundled information being inlined would efficiently provide support for compiler needed information in bundling a plurality of request pre-gathered knowledge for compilation without having to reach the granularity level in dynamically fetching informational resources-- per loading instance of bytecode class/method - implicated with each compilation request taken individually – information for one bytecode set at a time - by the runtime JIT in view of Holzle's dual interpreter/compilation scheme (see Fig. 6).

***Conclusion***

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 ( for non-official correspondence - please consult Examiner before using) or 571-273-8300 ( for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

November 02, 2010